Trees

ZyBook: Chapter 6

(picture)

Vocab

- root
- degree
- parent
- child
- leaf
- siblings
- ancestors

Vocab

- descendants
- subtree
- level
- height of a tree
- depth of a node

Binary Tree

- Any tree where a node has at most two children
- Very weak definition
- No one uses this

Binary Search Tree

- Definition:

- Key / Value pair

- Why is this useful?

Define a BSTNode

Build a BST

- Key/Value: 201/Doug, 202/Chadd, 203/Shereen, 211/Chris
  - What issues do we see?

Walk the tree

- Pre order
- In order
- Post order

bstSearch

bstInsert

- Write an algorithm for bstInsert.
- What is the worst case computing complexity of your algorithm? Why?
- Write the C function bstInsert.

FindLevel

- Write a C function bstFindLevel that returns the level of a node in a BST.

Recursion!

- A function that calls itself!

```
int foo(int x)
{
  if( x > 0 )
  {
    return 2 + foo(x-1);
  }
  return 0;
}


foo(2); // ???
```

Activation Records

- Each function adds one Activation Record
  - stack frame
- When the function terminates, the AR is popped off the stack

Recursion!

- Draw the activation records for foo(2);

```
int foo(int x)
{
  if( x > 0 )
  {
    return 2 + foo(x-1);
  }
  return 0;
}

int main()
{
  foo(2); // ???
}
```

Problem solving

- First step is to frame the problem in terms of itself.
  - a pattern
- Apply this pattern to create a recursive solution to the problem
- Divide a problem up into:
  - small unit of work
  - recursive call to do the rest of the work

Example

- A factorial is defined as follows:

  n! = n * (n-1) * (n-2) …. * 1;

- For example:

  1! = 1 (Base Case)

  2! = 2 * 1 = 2

  3! = 3 * 2 * 1 = 6

  4! = 4 * 3 * 2 * 1 = 24

  5! = 5 * 4 * 3 * 2 * 1 = 120

  Pattern? Small unit of work? Recursion?

Problems

- Write int factorial(int x)
- bstSearch()
- bstFindMaxDepth()